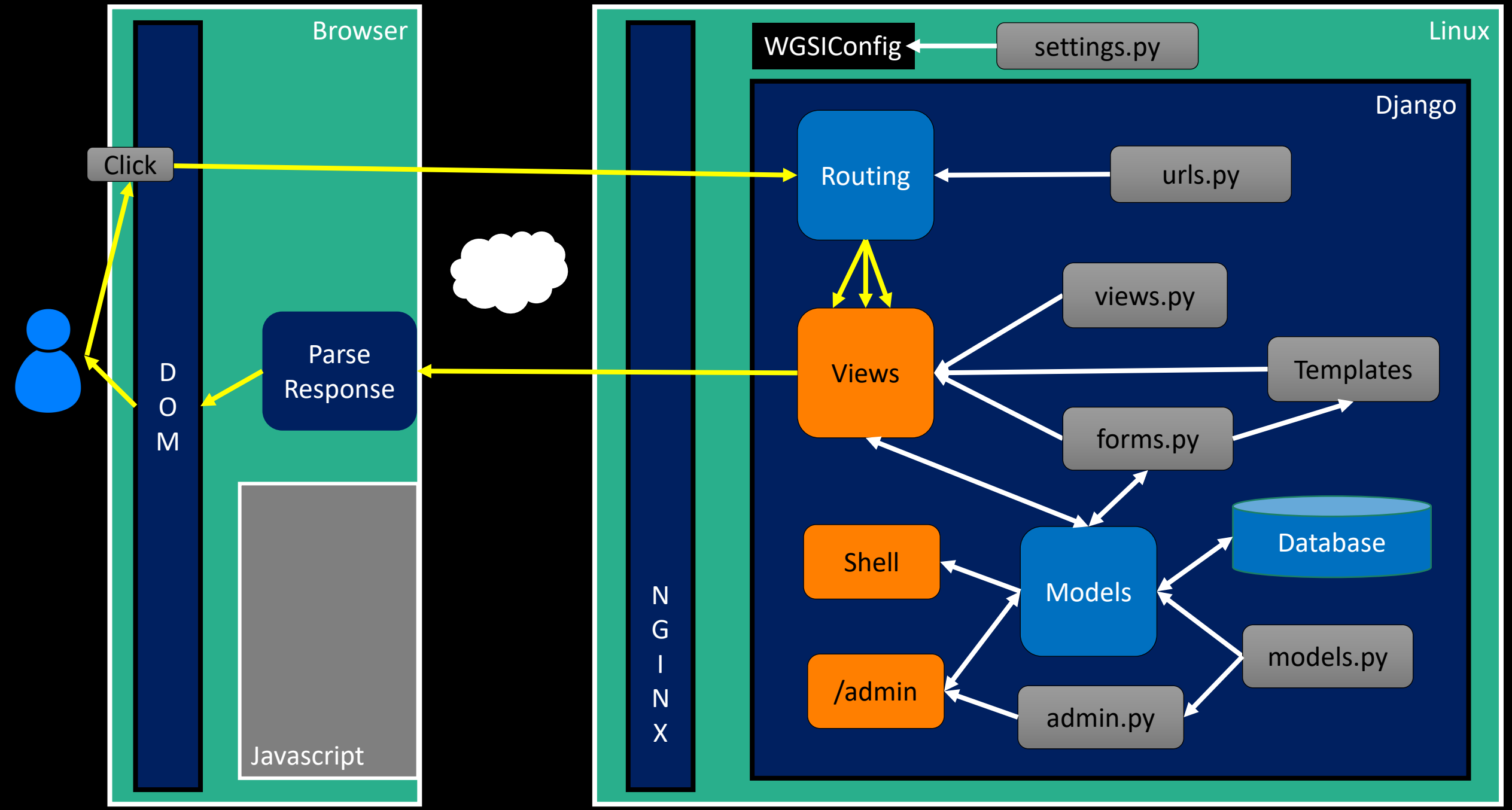
Table of Contents

This slide deck consists of slides used in 2 lecture videos in Week 3. Below is a list of shortcut hyperlinks for you to jump into specific sections.

Charles Severance
www.dj4e.com

Forms in Django

https://samples.dj4e.com/form/
https://docs.djangoproject.com/en/3.0/topics/forms/
https://github.com/csev/dj4e-samples/tree/master/form


Browser
Linux
WGSIConfig
settings.py
Django
Click
Routing
urls.py
views.py
Templates
Views
Parse
Response
forms.py
D
O
M
Shell
Database
Models
models.py
/admin
admin.py
N
G
I
N
X
Javascript

Django's role in forms (DRY)

Handling forms is a complex business. Consider Django's admin, where numerous items of data of several different types may need to be prepared for display in a form, rendered as HTML, edited using a convenient interface, returned to the server, validated and cleaned up, and then saved or passed on for further processing. Django's form functionality can simplify and automate vast portions of this work, and can also do it more securely than most programmers would be able to do in code they wrote themselves.

Django handles three distinct parts of the work involved in forms:

- preparing and restructuring data to make it ready for rendering
- creating HTML forms for the data
- receiving and processing submitted forms and data from the client

It is *possible* to write code that does all of this manually, but Django can take care of it all for you.

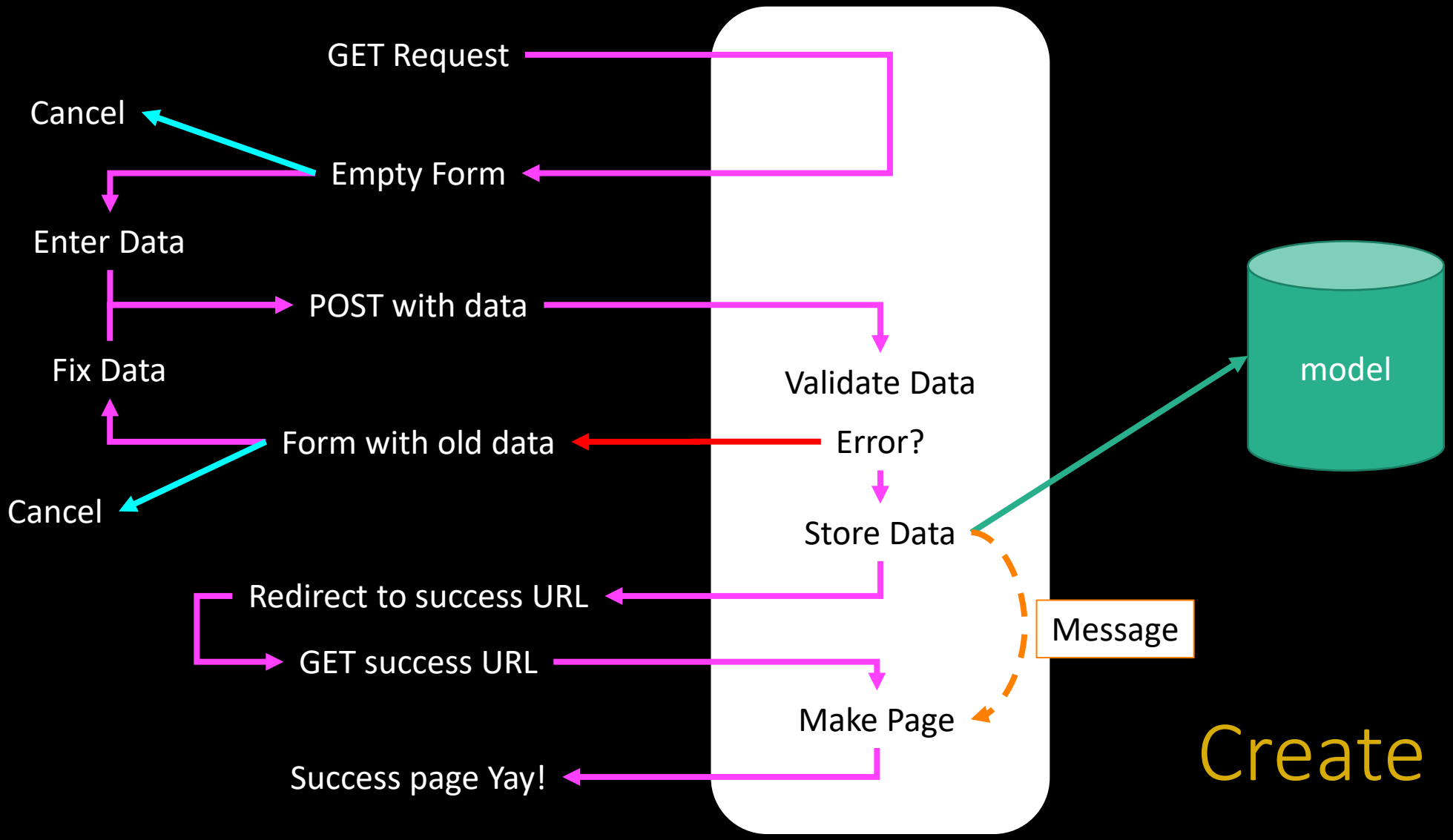
https://docs.djangoproject.com/en/3.0/topics/forms/#django-s-role-in-forms

It takes a lot of CSS to make forms pretty

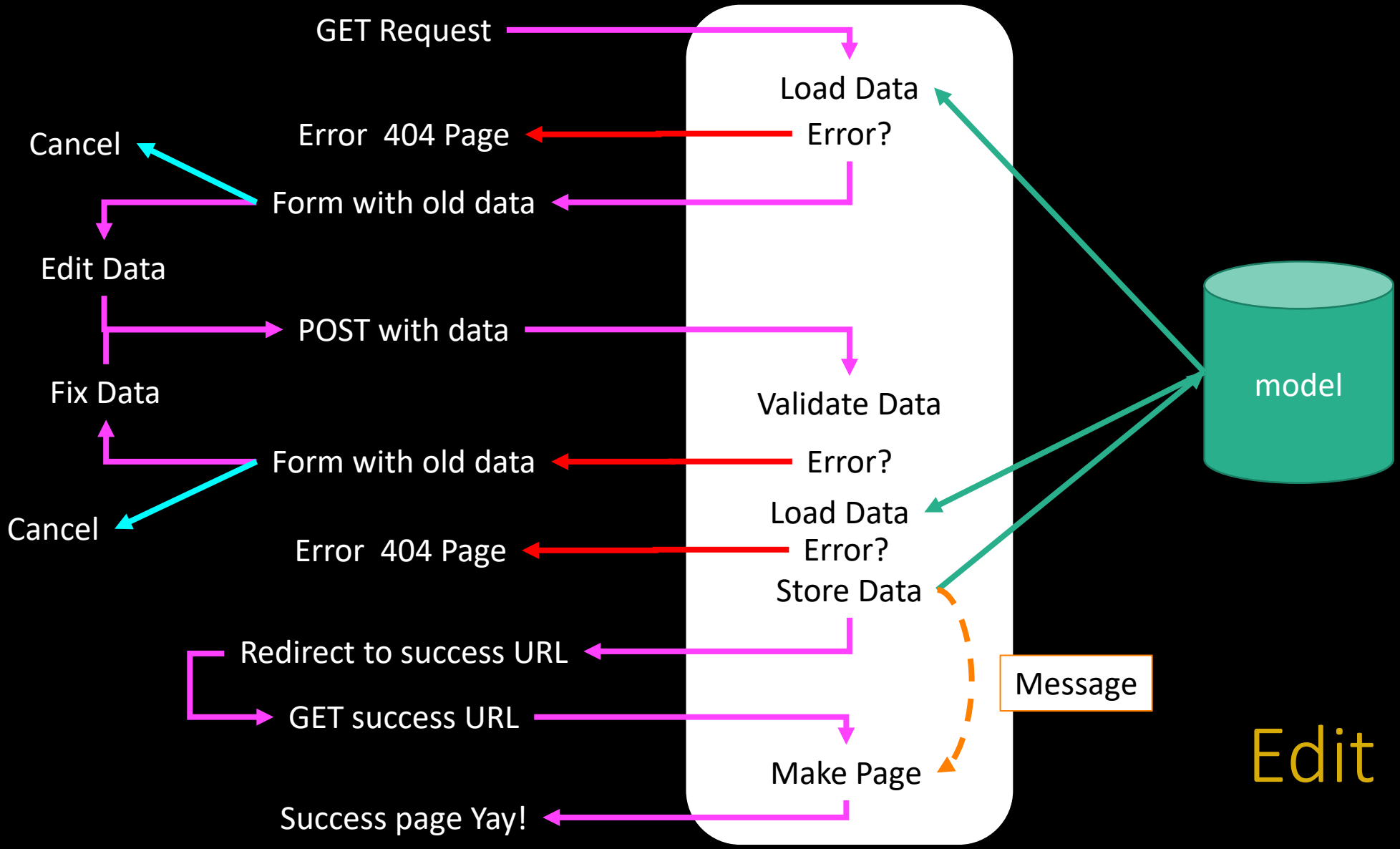
https://samples.dj4e.com/accounts/login/

Form Handling Flow is Actually Complex

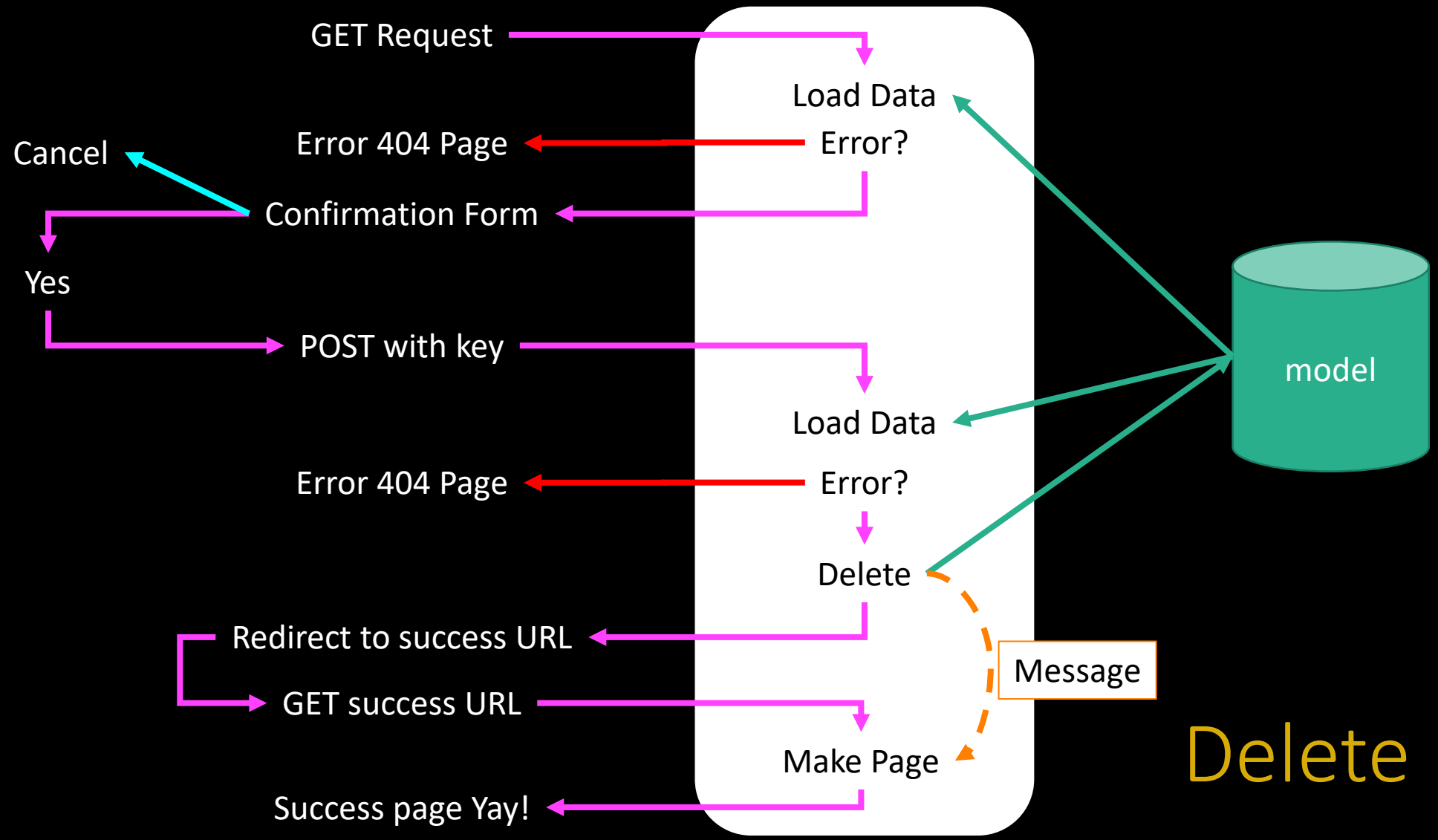
- Create
  - Produce empty form, check post data for validity, re-display form with errors if necessary, add the data to the database, and redirect the user to a success page with a success message
- Update
  - Load old data, form with old data, check post data for validity, re-display form with errors if necessary, update the data to the database, and redirect the user to a success page with a success message
- Delete
  - Load old data, produce confirmation page with a POST form, receive the post data, delete the record, and redirect the user to a success page with a success message



Create Form Flow



Edit Form Flow



Delete Form Flow

Django forms act as "glue"

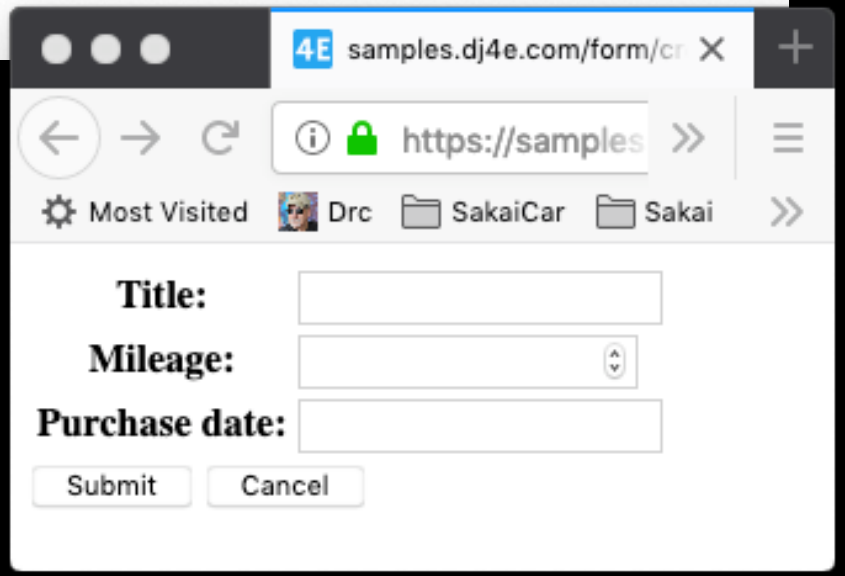
- Generate the necessary HTML to send to the browser
  - Allow for consistent look and feel across all the forms in an application
- Receive the POST data coming back from the browser
- Validate the incoming POST data and produce HTML for an error screen if necessary
- Move the data from the form into a model and then store it in the database automatically

https://docs.djangoproject.com/en/3.0/topics/forms/#django-s-role-in-forms

dj4e-samples/form/forms.py

```
from django import forms
from django.core.exceptions import ValidationError
from django.core import validators

class BasicForm(forms.Form):
    title = forms.CharField(validators=[
        validators.MinLengthValidator(2, "Please enter 2 or more characters")])
    mileage = forms.IntegerField()
    purchase_date = forms.DateField()
```

https://samples.dj4e.com/form/create



A simple form

Dumping a form object

https://samples.dj4e.com/form/example

dj4e-samples/form/forms.py

```
class BasicForm(forms.Form):
    title = forms.CharField(validators=[
      validators.MinLengthValidator(2, "...")])
    mileage = forms.IntegerField()
    purchase_date = forms.DateField()
```

dj4e-samples/form/views.py

```
from form.forms import BasicForm

def example(request) :
    form = BasicForm()
    return HttpResponse(form.as_table())
```

```
<tr><th>
<label for="id_title">Title:</label></th>
<td><input type="text" name="title"
required id="id_title"></td></tr>
<tr><th>
<label for="id_mileage">Mileage:</label>
</th><td>
<input type="number" name="mileage" required
id="id_mileage">
</td></tr>
<tr><th>
<label for="id_purchase_date">
Purchase date:</label>
</th><td>
<input type="text" name="purchase_date"
required id="id_purchase_date">
</td></tr>
```

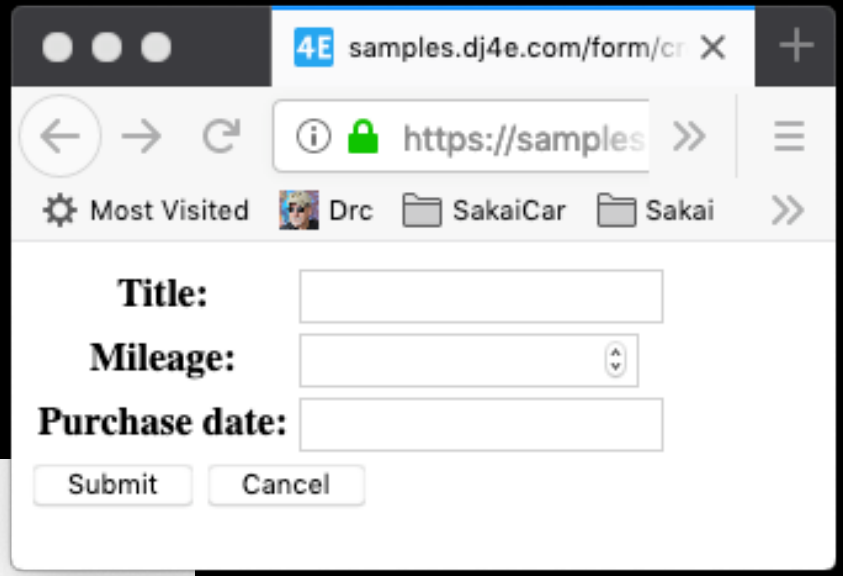
A form in a template

dj4e-samples/form/templates/form/form.html

```
<p>
  <form action="" method="post">
    {% csrf_token %}
    <table>
    {{ form.as_table }}
    </table>
    <input type="submit" value="Submit">
    <input type="submit"
    onclick="window.location='{% url 'form:main' %}' ; return false;"
    value="Cancel">
  </form>
</p>
```

A form in a template

https://samples.dj4e.com/form/create



dj4e-samples/form/views.py

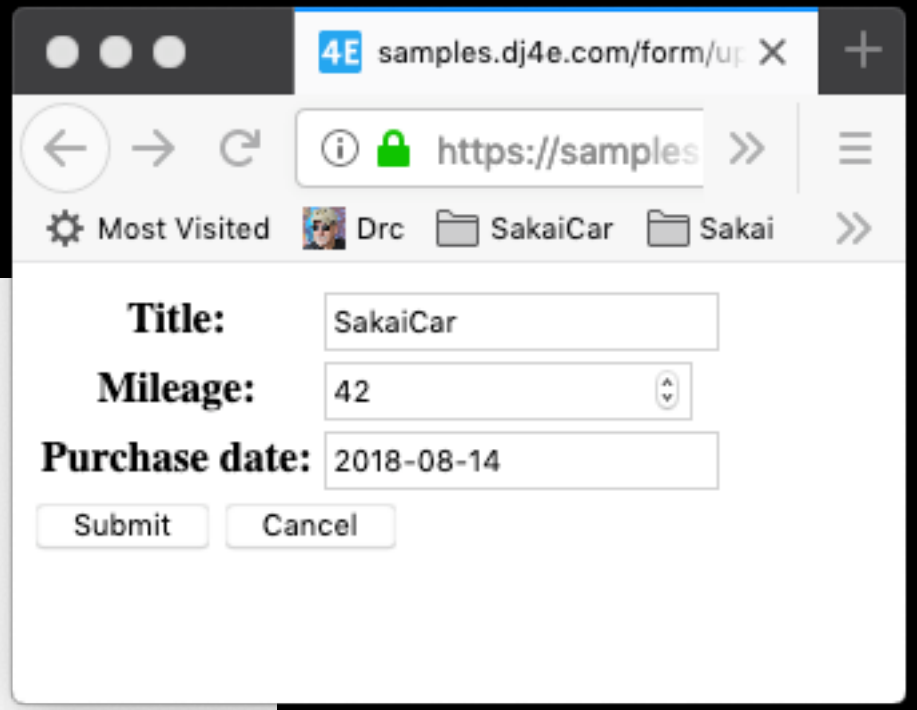
```
class SimpleCreate(DumpPostView):
    def get(self, request) :
        form = BasicForm()
        ctx = {'form' : form}
        return render(request, 'form/form.html', ctx)
```

Pulling existing data into a form

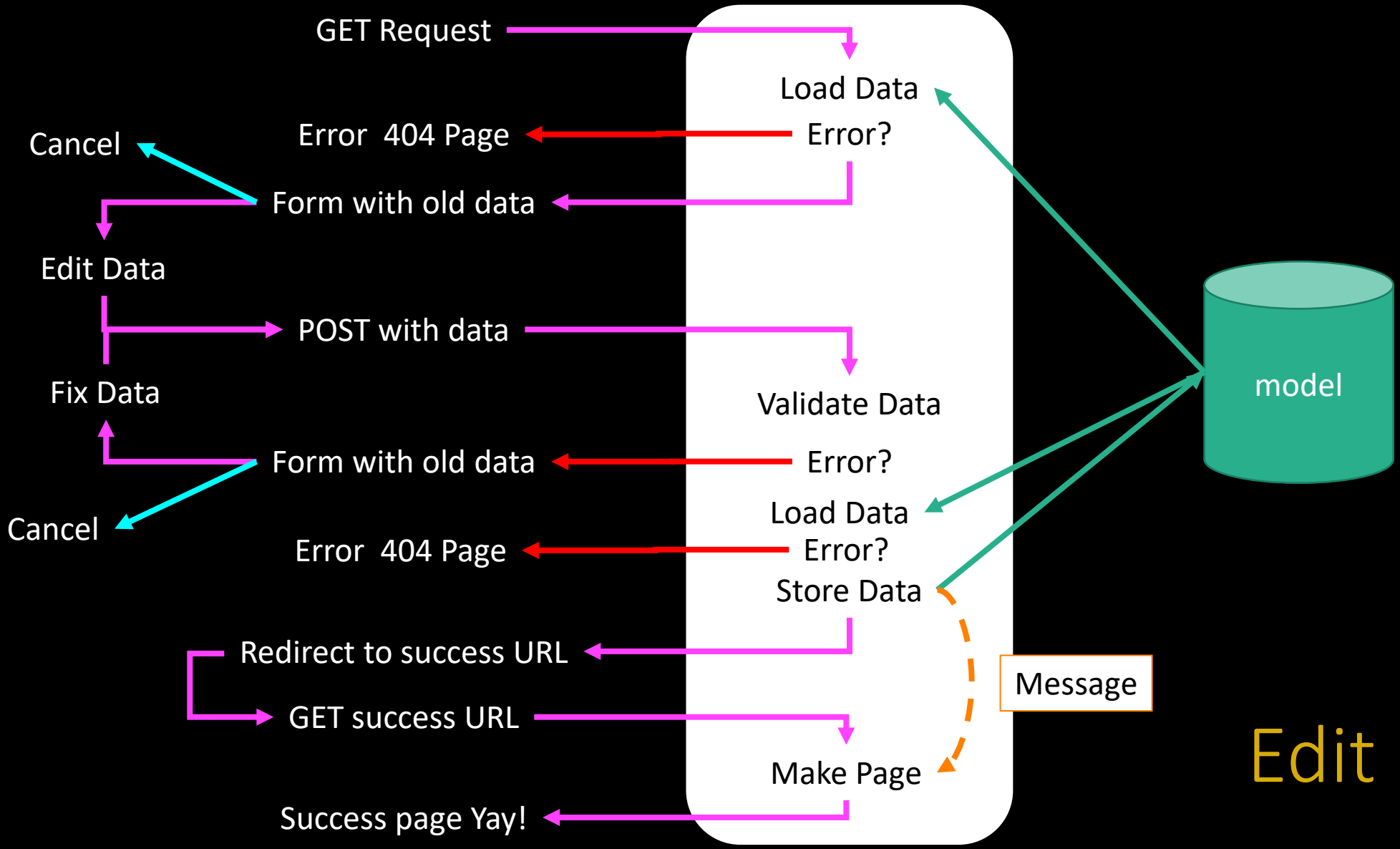
https://samples.dj4e.com/form/update

dj4e-samples/form/views.py

```
class SimpleUpdate(DumpPostView):
    def get(self, request) :
        old_data = {
            'title': 'SakaiCar',
            'mileage' : 42,
            'purchase_date': '2018-08-14'
        }
        form = BasicForm(old_data)
        ctx = {'form' : form}
        return render(request, 'form/form.html', ctx)
```



Data Validation in FORMS

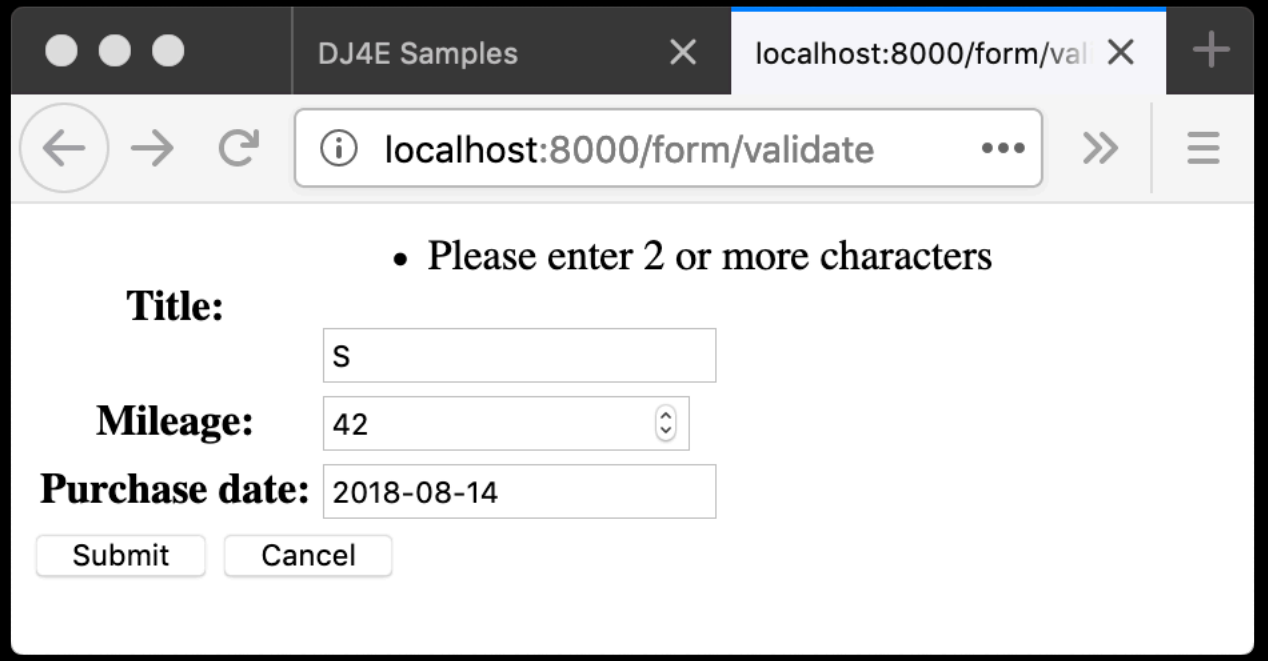


Edit Form Flow

Form Data Errors

- Sometimes there are validation rules when you are filling out a form.
- When you submit the form, the view code checks the data to see if there are errors
- If there are errors, data is not saved and the user is notified and usually given a chance to edit and resubmit

https://samples.dj4e.com/form/validate



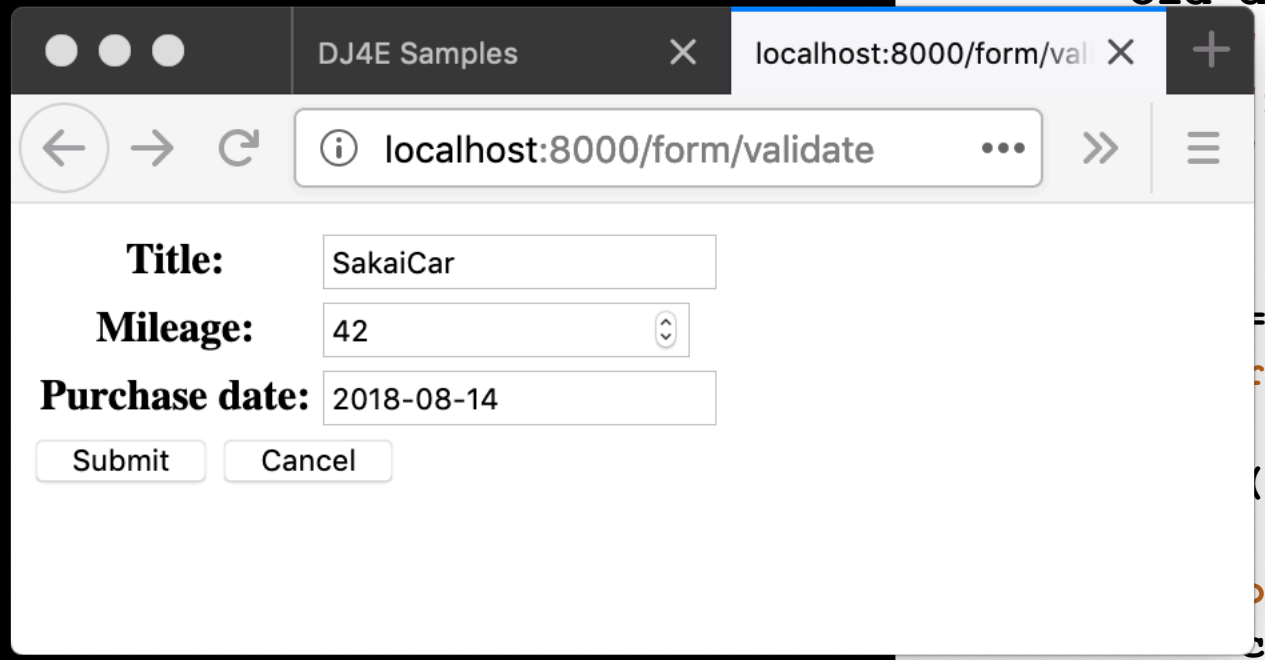
Django form validation

dj4e-samples/form/forms.py

```
class BasicForm(forms.Form):
    title = forms.CharField(validators=[
      validators.MinLengthValidator(2, "Please enter 2 or more characters")
    ])
    mileage = forms.IntegerField()
    purchase_date = forms.DateField()
```

https://docs.djangoproject.com/en/3.0/ref/validators/

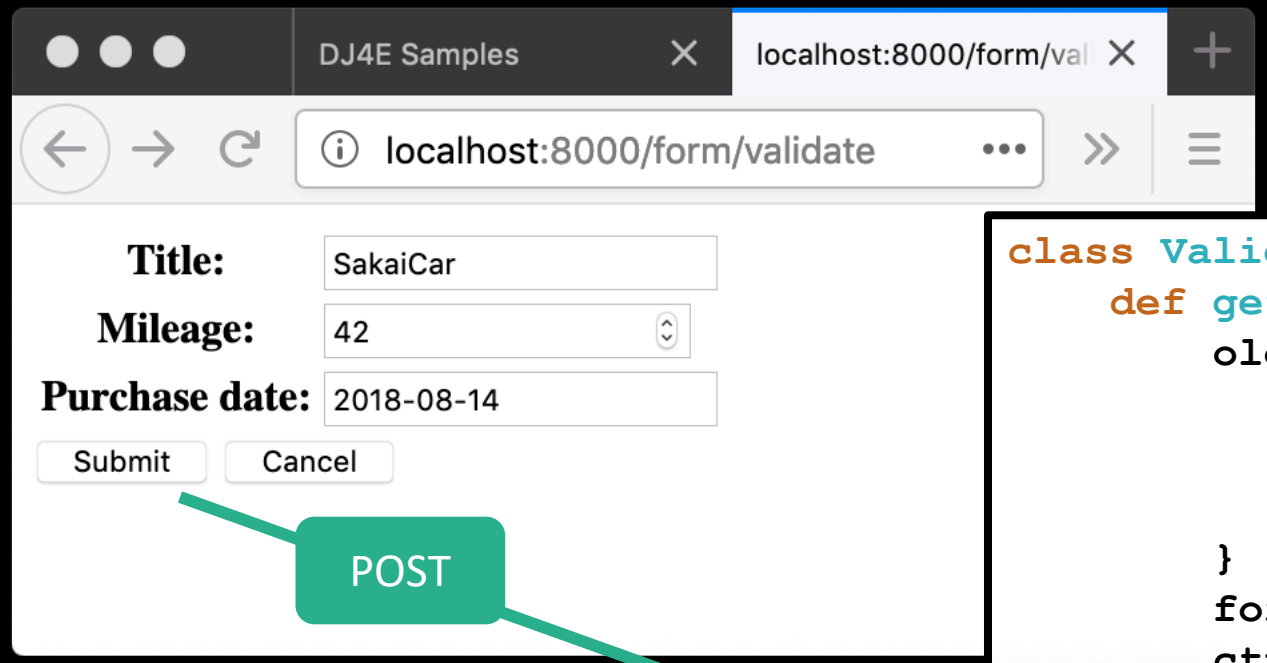
GET

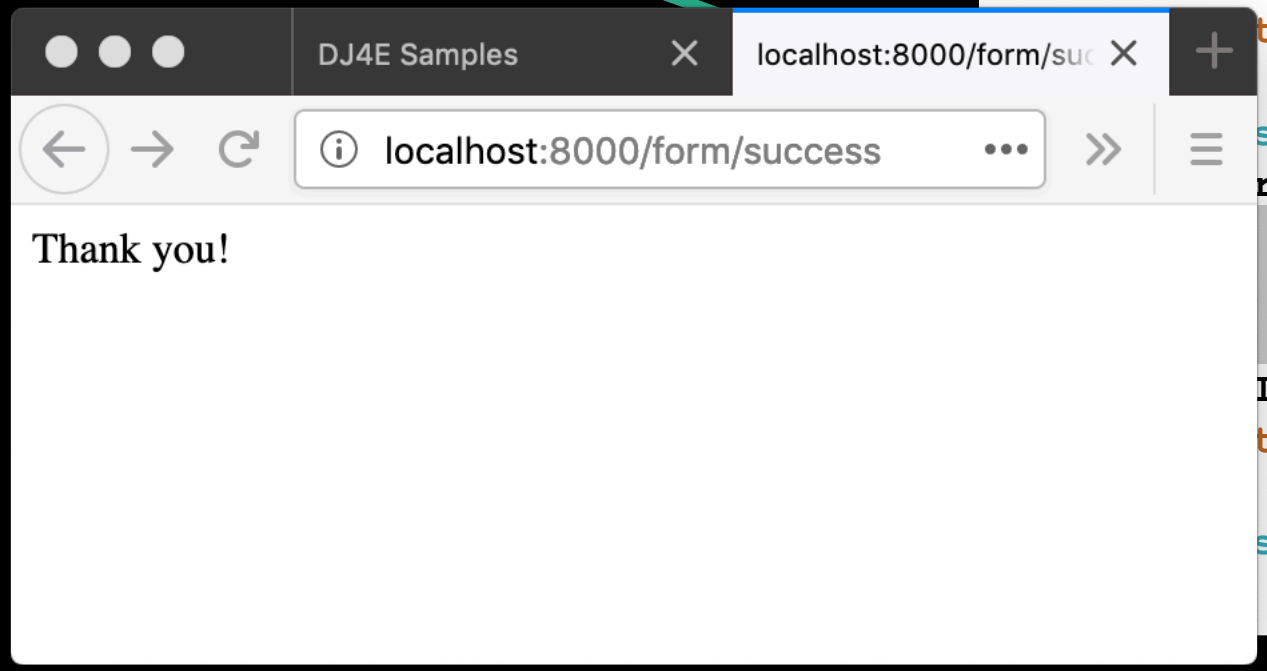



```
class Validate(View):
    def get(self, request) :
        old_data = {
            'title': 'SakaiCar',
            'mileage' : 42,
            'purchase_date': '2018-08-14'
        }
        form = BasicForm(initial=old_data)
        ctx = {'form' : form}
        return render(request, 'form/form.html', ctx)

    def post(self, request) :
        form = BasicForm(request.POST)
        if not form.is_valid() :
            ctx = {'form' : form}
            return render(request, 'form/form.html', ctx)
        # Save the Data
        return redirect('/form/success')

def success(request) :
    return HttpResponse('Thank you!')
```

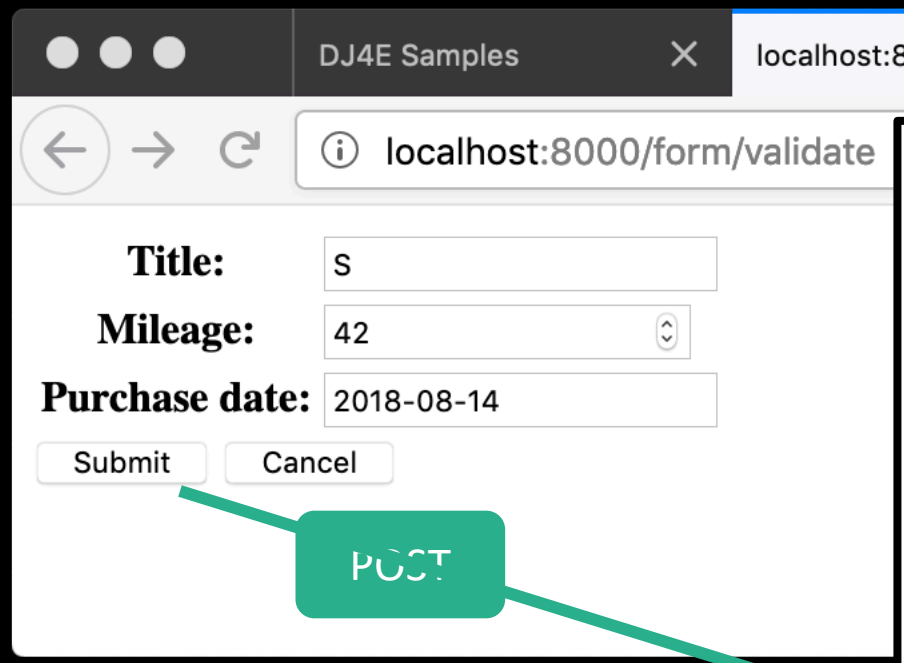
dj4e-samples/form/views.py

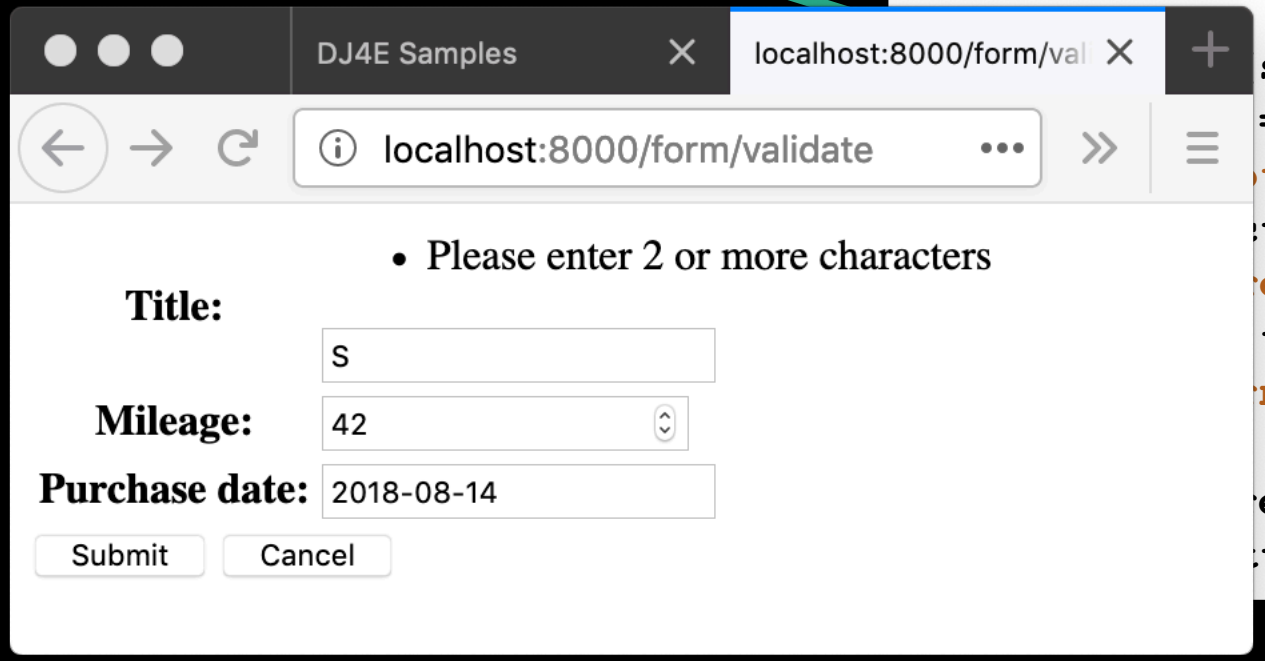
https://samples.dj4e.com/form/validate






```
class Validate(DumpPostView):
    def get(self, request) :
        old_data = {
            'title': 'SakaiCar',
            'mileage' : 42,
            'purchase_date': '2018-08-14'
        }
        form = BasicForm(initial=old_data)
        ctx = {'form' : form}
        turn render(request, 'form/form.html', ctx)

    st(self, request) :
    rm = BasicForm(request.POST)
     not form.is_valid() :
      ctx = {'form' : form}
      return render(request, 'form/form.html', ctx)
    If there are no errors, we would save the data
    turn redirect('/form/success')

    s(request) :
     HttpResponse('Thank you!')
```






```
class Validate(DumpPostView):
    def get(self, request) :
        old_data = {
            'title': 'SakaiCar',
            'mileage' : 42,
            'purchase_date': '2018-08-14'
        }
        form = BasicForm(initial=old_data)
        ctx = {'form' : form}
        return render(request, 'form/form.html', ctx)

self, request) :
= BasicForm(request.POST)
t form.is_valid() :
tx = {'form' : form}
eturn render(request, 'form/form.html', ctx)
there are no errors, we would save the data
n redirect('/form/success')

equest) :
tpResponse('Thank you!')
```

Models + Forms

Crossing the streams

https://docs.djangoproject.com/en/3.0/topics/forms/modelforms/

Form Structure is similar to Model Structure

dj4e-samples/form/forms.py

```
class BasicForm(forms.Form):
    title = forms.CharField(validators=[
      validators.MinLengthValidator(2, "...")])
    mileage = forms.IntegerField()
    purchase_date = forms.DateField()
```

dj4e-samples/form/models.py

```
class Cat(models.Model) :
    name = models.CharField(
            max_length=100,
            validators=[MinLengthValidator(2, "..")]
    )
    breed = models.CharField(max_length=100)
    comments = models.CharField(max_length=100, blank=True)
```

We can derive a form from a model

dj4e-samples/form/models.py

```
class Cat(models.Model) :
    name = models.CharField(
            max_length=100,
            validators=[MinLengthValidator(2, "..")]
    )
    breed = models.CharField(max_length=100)
    comments = models.CharField(max_length=100, blank=True)
```

dj4e-samples/form/forms.py

```
class CatForm(ModelForm):
    class Meta:
        model = Cat
        # fields = ['name', 'breed', 'comments']
        fields = '__all__'
```

dj4e-samples/form/views.py

```
class CatCreate(View):
    def get(self, request) :
        form = CatForm()
        ctx = {'form' : form}
        return render(request, 'form/form.html', ctx)

    def post(self, request) :
        form = CatForm(request.POST)
        if not form.is_valid() :
            ctx = {'form' : form}
            return render(request, 'form/form.html', ctx)

        # Save the form and get a model object
        newcat = form.save()
        x = reverse('form:main') + '#' + str(newcat.id)
        return redirect(x)
```

dj4e-samples/form/views.py

```
class CatUpdate(View):
    def get(self, request, pk) :
        oldcat = get_object_or_404(Cat, pk=pk)
        form = CatForm(instance=oldcat)
        ctx = { 'form': form }
        return render(request, 'form/form.html', ctx)

    def post(self, request, pk) :
        oldcat = get_object_or_404(Cat, pk=pk)
        form = CatForm(request.POST, instance=oldcat)
        if not form.is_valid() :
            ctx = {'form' : form}
            return render(request, 'form/form.html', ctx)

        editcat = form.save()
        x = reverse('form:main')
        return redirect(x)
```

Summary

- Django forms
- Form Validation
- Models and Form

Acknowledgements / Contributions

Initial Development: Charles Severance, University of Michigan School of Information

Insert new Contributors and Translators here including names and dates

Continue new Contributors and Translators here

Additional Source Information

- Portions of the text of these slides is adapted from the text www.djangoproject.org web site. Those slides which use text from that site have a reference to the original text on that site. Django is licensed under the three-clause BSD license.